Toward Specification and Composition of BoxScript Components

H. Conrad Cunningham, Yi Liu, and Pallavi Tadepalli
University of Mississippi, University, MS 38677

Abstract
BoxScript is a Java-based, component-oriented programming language whose design seeks to address the needs of teachers and students for a clean, simple language. This paper briefly describes BoxScript and presents the authors’ preliminary ideas on specification of components and their compositions.

BoxScript
Goal: To develop a simple, Java-based, component-oriented language that enables students to “think in components” and build simple systems.

BoxScript is a Java-based, component-oriented programming language whose design seeks to address the needs of teachers and students for a clean, simple language. This paper briefly describes BoxScript and whose design seeks to address the needs of teachers and students for a clean, simple language. This paper briefly describes BoxScript and presents the authors’ preliminary ideas on specification of components and their compositions.

Figure 1. Abstract boxes CalendarAbs and DateAbs

Abstract box CalendarAbs
{ provided interface DayCal Dc; 
  //Dc is handle of interface DayCal
}

Figure 2. Atomic boxes Calendar and Date

Box Calendar implements CalendarAbs
{ provided interface DayCal Dc; }

Box Date implements DateAbs
{ provided interface Display Dis; 
  required interface DayCal DayC;
}

Figure 3. Composition

abstract box BuildCalendarAbs
{ provided interface Display D;
  //D is handle of interface DayCal
}

Figure 4a. Abstract box BuildCalendarAbs
box BuildCalendar implements BuildCalendarAbs
{ composed from DateAbs boxD, CalendarAbs boxC; 
  //boxD is box handle for DateAbs and boxC is box handle for CalendarAbs
  provided interface Display D from boxC.Dis;
  connect boxC.DayC to boxD.Dc;
}

Figure 4b. Compound box CalendarAbs

Specification
An interface information model consists of pair (V,J):
- V is a set of abstract variables representing state of component instance
- J is an invariant that must hold in all client-visible states

Preconditions and postconditions specify the semantics of operations. Box interface x satisfies interface y when x provides at least the operations of y and the corresponding operations of x and y have equivalent meanings.

For a box B, let I(B) be its box invariant, C(B) be the coupling invariant that ties it to the interface information models, and prov(B) be the provided interfaces. For any box B, it must be the case that:

\((\forall p \in \text{prov}(B) : I(p)) \land C(B) \Rightarrow I(B)\)

An atomic box must implement its provided interfaces as a cluster of Java classes. All of its provided interfaces must have the same information model (V,J).

Figure 5. Interface x with operation m satisfies interface y. I(x) is the invariant for x, pre(x,m) is the precondition for operation m on interface x, post(x,m) is the postcondition for operation m on interface x. C(x,y) is a coupling invariant that links the information models of interfaces x and y.

Current and Future Work
Current plans are to:
- implement BoxScript
- specify several examples
- relate to other formalisms
- integrate with JML or other tools
- investigate alternative required interface semantics (e.g., callbacks)
- develop decomposition techniques